Mobile Game Development
The Official GameSalad Guide to Game Development

Vice President, Careers & Computing: Dave Garza
Director of Learning Solutions: Sandy Clark
Senior Acquisitions Editor: Jim Gish
Managing Editor: Larry Main
Product Manager: Nicole Calisi
Editorial Assistant: Sarah L. Timm
Brand Manager: Kristin McNary
Market Development Manager: Deborah Yarnell
Senior Production Director: Wendy A. Troeger
Production Manager: Andrew Crouth
Senior Content Project Manager: Kathryn B. Kucharek
Art Director: Jack Pendleton
Media Editor: Debbie Bordeaux
Cover Image Raptor Storm
Image courtesy of GameSalad ©

© 2014 Delmar, Cengage Learning

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transmitted, stored, or used in any form or by any means graphic, electronic, or mechanical, including but not limited to photocopying, recording, scanning, digitizing, taping, Web distribution, information networks, or information storage and retrieval systems, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the publisher.

For product information and technology assistance, contact us at Cengage Learning Customer & Sales Support, 1-800-354-9706
For permission to use material from this text or product, submit all requests online at www.cengage.com/permissions.
Further permissions questions can be e-mailed to permissionrequest@cengage.com

Library of Congress Control Number: 2012944317
ISBN-10: 1-1336-0564-8

Delmar
5 Maxwell Drive
Clifton Park, NY 12065-2919
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at: international.cengage.com/region

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

To learn more about Delmar, visit www.cengage.com/delmar

Purchase any of our products at your local college store or at our preferred online store www.cengagebrain.com

Notice to the Reader
Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions. The publisher makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.
Mobile Game Development
play as you go

Key Chapter Questions

■ How have mobile devices *evolved over time* to become major game development hardware platforms?

■ What are some art, design, and programming *restrictions* associated with developing games for mobile devices?

■ What are some *hardware specifications* associated with different smartphone and tablet devices?

■ What are some popular game *genres* played on mobile devices, and how are mobile hardware features contributing to the development of brand new genres?

■ How can *GameSalad Creator* be used to create and publish mobile games for both iOS and Android devices?

■ How will mobile game development continue to evolve in the *future*?

Mobile game development is tempting. From the seasoned developer tired of spending 3-5 years polishing and revamping the look-and-feel of an AAA game, to the bright-eyed bushy-tailed game designer itching to delve into a dream—mobile games offer a wealth of opportunity. As the mobile industry continues to change the way games are played and produced, throngs of developers are jumping on the bandwagon and releasing games for mobile devices—particularly those running either iOS or Android operating systems. GameSalad Creator is the perfect tool for this growing segment of the industry; let’s take a look at how it can be used to develop and publish mobile games on smartphones and tablets using the iOS and Android native operating systems.
Games that could be carried around and played “on the go” have been popular throughout history. Well before electronic games were ever developed, we relied on cards, dice, checkers, chess pieces, boards, and other “analog” materials whenever there was a need for fun. As technology began to advance enough during the later 20th century, dedicated electronic toys emerged—and handheld electronic systems such as Mattel’s *Electronic Quarterback*, *Merlin*, and Atari *Touch Me* quickly rose in popularity. This marked the dawn of the mobile game industry.

### Early Handheld Systems

Early handheld games were very simple and relied more on custom-built circuit boards than on the software commonly used today. In fact, single-game electronic devices such as the successful *Tamagotchi* keychain virtual pets still exist. However, these devices were soon overshadowed by handheld systems that allowed users to play multiple games—often using some form of solid-state storage, which most know as cartridges.

In 1989, Nintendo released the original Game Boy, which was followed shortly afterward by the Atari Lynx and Sega Game Gear. Although the other devices had color screens, it was the Game Boy that survived through multiple versions and is still popular to this day—eventually evolving into the dual-screen Nintendo DS. The DS’s lower touchscreen changed the way handheld games were played—and its successor, the DSi, added cameras and the ability to download games from an online store. The most recent model, the 3DS, supports 3D on the top screen using autostereoscopy (without requiring special glasses). Sony has also been a player in the handheld market, entering the field with the PSP (PlayStation Portable) in 2006 and most recently launching both the PlayStation Vita (dedicated handheld game console) and the Xperia Play (phone/game hybrid). Early handhelds all relied on variations of classic analog controls: buttons, joysticks, D-pads, and similar options. The newest generations of these handhelds (PS Vita, 3DS) have incorporated reliable touchscreen technology as well—allowing for an even broader range of gameplay styles.
Personal Digital Assistants

While the handheld segment of the mobile industry continued to advance—and well before wireless Internet access—personal digital assistants (PDAs) positioned as productivity tools were developed in parallel by electronics giants Sony, Casio, Palm, and Hewlett-Packard. The earliest PDAs didn't do much more than store an address book—not unlike Casio Databank watches. However, they improved as technology continued to advance—and devices running Palm OS and Microsoft's Pocket PC OS eventually became very popular. These devices supported many downloadable programs and games found in early app stores that sported larger screens and greater depth of functionality that would later emerge in the first generation of "smartphones."

At the same time, cell phone technology was also advancing. Although the earliest cell phones were "just phones," games began to show up just as the technology advanced enough to accommodate them. Nokia included Snake on its phones as early as 1997. Snake-style games were first introduced during the arcade era—and as such, the gameplay style was immediately familiar to the general public. Couple this familiarity with the ease of converting the control scheme to a mobile phone, and the game became a runaway hit on mobile devices. Color screens and other advances caused a surge in game development for cell phones in the early 2000s, but each game had to be purchased directly from the associated "carrier deck." Many consumers were unaware that these games existed—and those who did often didn't know to purchase them! Development was also extremely difficult due to the wide variety of phone hardware available and was primarily done using either Java ME or Qualcomm's Brew (C++).
N-Gage: A Handheld Hybrid Misstep

In an attempt to create a hybrid game/phone system, Nokia released the N-Gage (2003) and the N-Gage QD (2004). Despite a widely anticipated release, the N-Gage didn’t meet sales expectations—in part because it didn’t work well as a phone; its uncomfortable “taco” shape forced it to protrude out from the ear at an odd angle, and changing games required removing the plastic cover and battery compartment. Nokia later used the “N-Gage” brand to offer downloadable games, instead of the cartridges used by previous systems. Despite its shortcomings, the N-Gage proved to be a somewhat prophetic development—being the first to market with a number of developments that were perfected in today’s crop of cutting-edge smartphones.

In 2003, Palm released the Treo 600—the first real attempt to combine PDAs with cell phones. This kick-started the development of smartphones—but it was Apple’s iPhone, with its readily available and heavily marketed app store, that successfully convinced consumers to start downloading multiple apps (including games) to their phones. Smartphones available today include models running Windows Phone, BlackBerry, and Nokia’s Symbian OS—but iOS and Android devices are at the top of the heap.

iOS & Android Development

The iPod, released in 2001, was primarily a music player—but since it was a handheld device based on a microchip, its engineers just had to include some games. (For as long as the game industry has been in existence, games as “Easter eggs” have been included in hardware and software!) The first iPod included the game, Brick (originally created by Steve Wozniak), as an Easter egg—and more games were featured on later iPods. The addition of color and the creation of the iPhone and iPod Touch in 2007 (a touchscreen version with a much larger screen; an iPhone without the phone functionality) resulted in the iPod gradually becoming as much of a game device as a music system. A year after Apple launched the first iOS devices, Google released its open source OS—Android. With its own app store, Android is quickly catching up to Apple as far as the number of available apps and games.

In early 2010, Apple introduced the iPad—a larger tablet running iOS. Tablet devices had been around for years running Microsoft Windows but never had a very large following. However, the iPad was such a success that it triggered the development of many other tablets running Android OS. Even major book retailers now have their own Android-based tablets, such as Amazon’s Kindle Fire and Barnes & Noble’s Nook.
Operating Systems: iOS vs. Android

Understanding the differences between the iOS and Android operating systems might help you decide which one to initially focus on. One difference involves hardware. Keep in mind that iOS only runs on a limited number of iPhone, iPad, and iPod Touch models. Due to this, you are also guaranteed a certain amount of compatibility. Android runs on a much wider range of hardware, making it much harder to guarantee that your program will run on all of them. Applications using iOS and Android are developed very differently from one another. Consider that iOS applications are generally built using Objective C—a deceptively named language, since “C” is for “Cocoa” and there is very little resemblance to C/C++. Objective C was designed by Apple to be easier for non-programmers and those unfamiliar with the C syntax of most other languages. Android applications are built using Java, which is more familiar to most programmers. This means that unless you’re generating a game through a third-party program such as Creator, developing a cross-platform app means writing the same app twice in different languages.

Mac or PC?

Since any iOS game has to be run through Xcode on its way to the Apple App Store, iOS titles will need to be published with the Mac version of Creator. The Windows version of Creator can be used for development, but it will be necessary to export to the Mac version to actually publish the game for iOS.
Although there are development differences between iOS and Android, it can be argued that the biggest distinction between them is actually the way each of their associated marketplaces functions. Apple maintains tight control of all apps that are released on the iTunes store. Adult content, for example, is not allowed in an iOS app—and neither are products that might contain “hate speech.” This is reminiscent of the rules developed by Nintendo in the early days of console gaming to make its Nintendo Entertainment System (NES) platform more family-friendly.

Most Android app stores have no such restrictions. This has practical consequences beyond simply the game’s rating. The vetting process for iOS apps can be long and frustrating. Since time is money, it can be expensive for a small developer to wait for Apple’s approval process. Quick, nimble development houses may be more attracted to the more straightforward release process for Android apps. Although there are some “walled garden” Android stores popping up (most notably Amazon’s Appstore), they do not yet represent a significant percentage of the market.

Multiple Platforms . . . and More!

I use Creator to make a number of games from the ground up, and I always design my games for both iOS and Android. I usually start with iOS and then build an Android version. The power I see with Creator is the cross platform ability; it also allows me to get my ideas made and out there quickly so that I can move on to my next idea or client gig.

Caleb Garner (Game Producer, Part12 Studios)

With Creator, you can make one game and publish it to both iOS/Mac and Android (including Kindle Fire, Nook, and Android marketplace). It’s wonderful and easy to use—and it allows you to publish to all the latest platforms and monetize your app in more than one store.

Jenny McGettigan (Owner, Beansprites LLC and Hamster Wheel Studios)

I’ve been developing and publishing for iOS (iPhone and iPad) as well as MacOS. For both platforms, Creator is an easy to use but powerful tool for extremely fast app creation.

Thomas Wagner (Owner, Gamesmold)
Programming: Xcode vs. Eclipse

If you are not using a separate tool such as Creator to develop your games, you will generally be developing with either Xcode or Eclipse. Even if you are already working with a licensed set of programming tools, you may need to use either Xcode or Eclipse as the final step before publishing if you want to include new features (e.g., analytics, third-party add-ons).

Xcode is an integrated development environment (IDE) that was developed by Apple specifically for iOS and Mac development. It’s free for OSX users and includes an emulator—a tool that simulates what the game will look like on the actual device. This emulator will load automatically when you run your game from Xcode.

Xcode for Android

Although Xcode’s main function is as an iOS development platform, some have had success using Xcode for Android platform development. See David Janes’ Code Weblog at http://code.davidjanes.com/blog/2009/11/20/how-to-use-xcode-for-android-projects/ for more information.

Eclipse is a free, open-source IDE that can be used with a number of programming languages—including Java, C++, Ruby, Python, and Perl. Based on plug-ins (and relying on them for most of its functions), Eclipse is often used for Java programming. The number of plug-ins available makes it possible to create software for a huge variety of environments and architectures. The Android plug-ins mesh well with Eclipse and install an emulator that will run automatically for testing.
IDEs such as Xcode and Eclipse will form the basis of your toolset when developing programs for mobile devices; being familiar with them will give you much more flexibility as a designer and developer—even if you choose to use pre-fabricated tools for prototyping. These IDEs are by no means exclusive to mobile development; they are commonly used in both Windows and Mac development processes and to develop a broad range of non-mobile products.

Eclipse, which can be downloaded for free through the Android Developers site, is a well-supported integrated development environment (IDE) that can be used for the Android platform.

Android emulators provide the functionality of multiple devices with the same operating system, but with different processor specifications or hardware.

**Eclipse Game Development Community**

For every tool, you will find a community of developers who dedicate their time and resources to expanding and improving on it. Often, the strength of a tool lies more in the support it receives from its community contributions (tutorials, discussion groups, forums, upgrades and improvements) than in the technology behind the tool itself. The Eclipse community can be found at http://www.touchofdeathforums.com/eclipse/
Game Engines & Drag-and-Drop Tools

One of the key differences between the previous and current generation of mobile phones and handhelds is the availability of pre-written game software. Until the iPhone hit the market in the early 2000s, all programming for mobile devices was custom (specific to the studio that worked on the title). The functionality of a particular studio’s code—and the speed and agility with which they could adapt it for new titles—was often a deciding factor when development contracts were handed out. This meant that small or even one-person studios were extremely rare, and the core of any team had to be centered around the programmer. However, drag-and-drop tools such as GameSalad Creator give designers and artists with limited programming skills the ability to prototype, design, and release their own products. This serves the dual purpose of allowing designers to experiment with new gameplay styles and ideas, as well as keeping a steady stream of new content flooding into the app store for all platforms.
Smartphones vs. Tablets

If you want to create a game that will work on both smartphones and tablets, you’ll need to focus heavily on the marked difference in screen size. It will be necessary to either offer an alternate graphical layout for the larger screen or create the graphics so that they still look good when stretched out on a larger screen. The best practice is to test the game extensively on different emulators or other devices. It’s also possible to develop different versions of the game for each type of device. Games for iPad only are available through the Apple App Store, and there are many Kindle Fire apps available through the Amazon Appstore.

The iPad screen is much larger than the iPhone 5 screen—with a resolution as large as 2048 x 1536 (iPad) vs 1136 x 640 (iPhone 5).

Community Love: Templates & Tools for Creator

GameSalad is supported by a very active and engaged community of developers. Many community members provide examples of their projects in progress—showing exactly how they have solved problems like object creation on the fly, or scoring and saving examples. They have built some very helpful Creator templates for shoot-em-ups, platformers, vertical shooters, tower defense games, and more.
Two dimensions are still much more popular than three in mobile applications, in contrast to games on non-mobile platforms. This is in part because the platforms are less powerful and don’t have the storage space for large 3D graphics (which means that 3D games run much more slowly). However, the rise of mobile games has inspired a rash of 2D games (including remakes of many old arcade- and early console-era 2D games); mobile players are not all “hardcore” gamers who demand 3D. In addition, older Android phones may not support OpenGL ES 2.0—a common platform for 3D on mobile devices.

There are tricks to prevent 2D art from seeming flat. Layered backgrounds, parallax motion, “2.5 dimensions,” and looping/tiling are all old-school game development tricks that help the developer get more use out of limited file storage space. Creator contains tools that add graphic complexity to a game project—making it ideal for a one- or two-person team.
Currently, 3D for mobile is gaining hold—and tools such as Unity are being used to specifically handle 3D development for mobile games. However, developing 3D mobile games requires the creation of 3D models in software such as 3ds Max or Maya—making the development process much more involved and time consuming.

**Necessity is the Mother of Invention**

The problems framed by the limited processing power of mobile devices are not new by any means. As the available processing power for desktop computers and game consoles increased, the need for developers of more traditional games to adhere to tight space constraints began to fall off. (“Best practices” still dictate clean and efficient use of resources, but the consequences of some “bit bloat” were often overlooked because they didn’t affect the final outcome as drastically.) With extremely limited download sizes and smaller development cycles (compared to desktops and consoles), mobile games required a re-imagining, and occasional reinvention, of older tricks—such as using a palette animation to give the appearance of motion to water in a scene or the flapping of a flag.

### Interface: Native iOS vs. Android

Both iOS and Android have built-in user interface elements that can be used to easily create simple interfaces. Standard user interface elements such as buttons, text fields, checkboxes, and drop down lists are handled natively by the operating system to maintain a consistent look and feel without having to write them from scratch. Interface Builder, a program that comes with Xcode, can be used to develop interfaces for iOS. Screens are generated by visually placing elements on the screen—and the tool generates an .xib file (essentially an .xml file that tells Xcode which objects to put where). Eclipse offers similar capabilities using .xml files. Interfaces can be created visually within Eclipse—or the .xml file can be manually edited to create the desired screen.

Source Apple Inc.

Source Google.

Interface Builder (left) and ADT Graphical Layout Editor (right) are used to develop interfaces for iOS and Android, respectively.
Project Setup

When you first set up your project, be sure to select the appropriate platform (in this case, Mobile). See Appendix for a tutorial on how to set up a project, and more!

Using the iOS/Android Viewers

Xcode can be used to install the Creator viewer onto iOS devices—allowing you to view your game on the actual device. This requires you to be registered as an iPhone developer and to generate a provisioning profile with Apple. A tutorial on how to install the Creator viewer is available at http://cookbook.gamesalad.com. This will allow you to easily test your game project on your mobile device.

The Android viewer requires you to be a little more familiar with the command line. The process is subject to change, but full up-to-date instructions for installing both versions can be found at http://cookbook.gamesalad.com/.

Case Study: Bumps

Creator has several strengths—including being fast and easy to use. I remember when I first looked into iOS for our hit PC game Bumps (we wanted a conversion from PC to iOS), some of the quotes we got back were in excess of $85k. I found GS and made it myself within 6 weeks! The physics in Bumps was something we were really pleased with, and changing them on the bumps was only a click away—so during testing, we used a variety of different settings in density, friction and “bounciness” until it felt right. Using only a small number of actors, we could create realistic curved platforms for the bumps to roll over—which was a really nice change from flat, boring rectangular platforms.

Darren Spencer (Owner, Utopian Games & Deep Blue Apps)
Marcin Makaj on Guidelines for Mobile Developers

Marcin Makaj might be one of the first GameSalad developers in Poland. Before discovering GameSalad Creator, Marcin didn’t have any experience in games and high technology. He has always been creative—enjoying drawing and building structures with Legos. He’s interested in computer games, economy, law, marketing, movies, and meditation. Marcin has a Master’s degree in Economics and is currently a law student. It has always been Marcin’s dream to make a game. His favorite game genre is the classic point-and-click adventures—first on an Amiga 600 and later on PC, Game Boy Color, iPhone, and iPad—which led to his adventure game series, The Moonwalls, developed with Creator.

Here are my guidelines for mobile developers:

1) Make one big game and one small game so that players have something to play while waiting for your next big title.

2) Try to keep your project and files organized. For example, use consistent naming systems for attributes, scenes, images, etc.

3) Make only games that you want to play.

4) Don’t expect to make a lot of money; just create and have fun!

5) There are great games with low sales; it all depends on good marketing.

6) Respect your players and customers; make the best games possible, answer their emails, and provide support.

7) Listen to feedback, read reviews, and always make improvements; it’s never too late to create a bestseller.

8) Don’t be afraid to invest money; if you treat development as a business, you will eventually make money.

9) Don’t rip off other games, steal ideas, or cheat Apple’s review policy; you will get sued or banned sooner or later.

10) Test, test, and test your game before releasing each update.

11) If you think that you have “the best idea ever” for a game, stop it! There are tens of thousands of other developers who think the same about their ideas—and most of them can be thrown in the trash.

12) Games with simple ideas are the best.

13) Love your work—and your players will love your work, too.
What's So Different about Mobile?

The mindset behind the design of most mobile games is different than with traditional desktop or console games. Mobile titles are not only designed for the “quick fix,” but they are generally much more permissive when it comes to gameplay styles and innovations. Advertising is permitted, even encouraged, by players who are more willing to download and play a free application that supports the developers through advertising than they are to pay for a product up front. So while the techniques of game design and development used in more traditional products still holds, the developers have a much broader range to play in. This is resulting in new and innovative gameplay types and control schemes, as well as creative advertising and marketing.

Mobile Design

Many different types of games are available in the app stores. Many older games from other platforms have been ported to mobile devices. Games such as Angry Birds are very popular—since people on the go like to play short, simple games for 3-5 minutes while waiting in line or otherwise passing the time. However, all types of games are available—from text adventures to full 3D first-person shooters to puzzle games … and everything in between!

Case Study: Quake Builder

The first game I made with Creator, Quake Builder was the first of many “Eureka!” moments. I learned my first bit of trigonometry (which I hadn’t touched in about 10 years) and made a platform rotate back and forth—with a lot of help from the GameSalad community. Quake Builder took around three months and two hours a day to develop. I made it available on the App Store for free; it received over 250,000 downloads in a week and was one of the US top 100 free games. I promptly added iAds to the game using Creator and started reaping the rewards!

Jon Draper (Director, Stormy Studio)
**Mobile Programming**

Mobile platforms are more limited and have less memory than most consoles or PCs. When programming for mobile devices, be aware of the limitations and make sure to optimize your game with this in mind. Best practices insist that any unused or unseen game elements be removed from memory. This prevents obvious issues like lag within the game and more insidious things like memory leaks or unusual conflicts between objects that are hard to debug later on.

**Mobile Art**

Games require assets—all of which will increase a game's total size. Since mobile games need to be downloaded before running, the space taken up by art needs to be as small as possible. Some mobile carriers even place limits on the size of apps that can be downloaded over their 3G/4G networks—and larger apps require a Wi-Fi connection. A game requires a number of art assets—including scrolling backgrounds, levels, particle effects, and sprite graphics. The small size of mobile screens is a major restriction on the types of art used in these games. The most successful mobile developers have been inspired by these limitations to incorporate vibrant, iconic graphics into their games.

---

**Powers of Two**

To avoid ugly artifacts in the final game, all artwork for an iOS device should be sized so that the pixel width and height are even numbers (i.e., divisible by two). This might seem like a strange rule, but it’s an important one! It is also generally better to use graphics loaded into memory at sizes that are powers of two: 8 x 8, 16 x 16, 32 x 32, etc. If a graphic’s width and height are even, the graphic will still be loaded into memory at the size of the next power of two. Therefore, a 20 x 20 image will be loaded as a 32 x 32 image—using more memory than you might have planned on. A 30 x 66 image will be treated as 32 x 128—which uses almost twice as much memory as it should! Note that images cannot exceed 2048 x 2048. It is also better to use .png files (although GameSalad Creator will convert images to this format automatically) at a resolution of 72—while maintaining aesthetic appearance.
Spooky Hoofs was my second endless runner game made with GameSalad Creator. Designed for a Halloween release, the player controls a stagecoach—dashing through eerie woods and spooky landscapes. To permanently have smooth, physically correct movements throughout the game, I tried to move the coach only by Creator’s physics engine. In order to do so, I couldn’t use some kind of pseudo scrolling; instead, I had to use a single, very large scene where there was enough room to really race around. The ground consists of seven blocks that keep on placing themselves outside the right border of the camera. This way, the coach can actually travel to the right and have the real physics working (with the camera smoothly following). Once the carriage has reached somewhat around x=10000 or so, I set everything (ground, coach, enemies, background, camera etc.) back to x minus 9000. This happens within a single frame and isn’t noticeable; this way, you get the feeling you are traveling endlessly.

Another challenging task was the creation of the carriage itself. With the horses, the coachman, the passenger, the shaking lantern, the wheels and all the other animated details, it consists of over 20 single actors (some visible and some not) that manage a very realistic, physically convincing real-time movement of the whole carriage. Most of the time, when it gets to detailed or complex animation, it is much more efficient to assemble an object of many separate actors rather than trying to create all animation simply by linear image sequences. Spooky Hoofs was “New and Noteworthy” worldwide on the iPhone and in the Mac App Store—and it is still is one of my best selling game apps so far.

*Thomas Wagner (Owner, Gamesmold)*
New Control Schemes

Smartphones have very different interfaces from most other devices. When accommodating control schemes for touchscreens, note that buttons need to be large enough to touch with a finger without accidentally hitting the buttons surrounding it—so it’s necessary to experiment with button sizes until you “get it right.” Creator was designed as a multi-platform product, so it includes the option to have an actor react to a finger touch, mouse click, or even a dramatic change in the orientation of the player’s smartphone using the built-in accelerometer (the sensor that tells the phone which end of the device is pointing up and whether it’s being rotated, shaken, etc.).

The touchscreen has developed new ways of interfacing, such as swiping a finger across the screen to scroll left or right. It is also common to use a two-finger interface for pinch-to-zoom, stretch-to-zoom, or pinch/stretch functionality: Many mobile apps allow stretch (moving fingers apart) to zoom in and pinch (moving fingers together) to zoom out.

Mobile games often take advantage of the accelerometer functionality, and Creator has a full suite of acceleration-based functions that you can use. Many games such as Pocket Labyrinth and Zombie Games! use the accelerometer as the main control scheme; by tilting the phone, it’s possible to control the direction you move in a game. Some games use the microphone as a controller—with players humming or blowing into it to navigate or control other actions. Other smartphone capabilities that can be used as control schemes include the camera, global positioning system (GPS), and gyroscope.

---

Tutorial: Up, Down & Sideways

Kimberly Unger (Chief Executive Officer, Bushi-go, Inc.)

Setting up a game to automatically flip over when the player’s smartphone is turned over—so that it will always be right-side up—is very simple in Creator. Inside the scene, select Scene/Attributes and expand Auto-rotate. For a Landscape game, check both “Landscape Left” and “Landscape Right.” For Portrait, check both “Portrait” and “Portrait Upside Down.” Creator will then flip the game automatically. Causing an object to move when the mobile device is tilted is a little more complicated:

1. Inside the Actor settings create a rule for each direction in Creator by clicking the “Create Rule” button on the top right. Then, choose “Attribute” and “Devices/Accelerometer” from the dropdown menu next to it. The attributes can be accessed via the “Attributes” tab in the browser at the bottom of the screen. Select and set an attribute after creating the rule and before adding the specifics of how the accelerometer reacts.
2. For a Landscape mode game, check “y” for left and right and “x” for up and down. For portrait, reverse this. To check for tilt, the first rule should check if this is greater than some small number (such as 0.01). If so, add a behavior inside the rule to handle this by finding “Accelerate” in the behavior library and dragging it inside the rule you have set up. Set the direction to 180 and acceleration to 200. In addition, set the dropdown for “Relative To: to scene.”

3. After creating one rule, a copy of it can be created by holding down Alt (Shift) and dragging the rule underneath it. Once this is done, the next rule can be created by changing “greater than” (>) to “less than” (<) and 0.01 to -0.01. You also need to change the acceleration behavior: Change “180°” to “0°” to reverse direction.

4. To move up and down, create new rules—but check the accelerometer “x” instead of “y.” The direction to accelerate will now be 90 and 270. (Consider basic geometry, where a circle is 360°—which will help you remember why the directions are 0, 90, 180, and 270!)

5. If you have implemented the automatic switching for “Landscape Left” and “Landscape Right”: After creating rules for all four directions, put them inside another rule that checks the Devices/Screen/Interface Orientation. Set this for “Landscape Left” and then copy the new rule and all its contents to create another one for “Landscape Right.” Switch the “Acceleration” directions for all the rules inside that one by 180°.
Making Mobile Development Fun

Creator makes mobile development not only a fast, robust experience, but one that is really enjoyable. Coding is not fun for me, I don’t enjoy typing out line after line of code. I’m a creative person by nature and enjoy seeing things that I’m creating happen visually. The development side of mobile gaming is notoriously tricky—with provisioning profiles, developer certificates, keystores, and other endless obstacles to overcome (besides learning the programming language). However, Creator takes that complicated process and makes it as easy as possible for beginners and advanced users alike.

Ace Connell (Managing Director, Mynameisace Ltd)
Publishing

Before beginning to develop an iOS game, check to make sure it fits Apple’s guidelines to maximize its chances of being accepted. To publish a game to the iTunes store, you must be an Apple Developer. Membership in the Developer community currently costs $99/year. Once an app has been submitted, it will go through a review process; it’s necessary to wait for approval and make any requested changes or fixes. Note: If your game is rejected, you can appeal that rejection to the App Review Board—a group of Apple employees who handle such appeals. The Android publication process is a bit complex, but it does not involve submission to a board for approval. If you want to publish your game for the Kindle Fire, you will need to upload your game to the Amazon Appstore. Membership is $99 a year.

With so many app stores available to consumers, publishers are in a rush to ensure that developers have all the necessary tools to serve up their content.

Apple & Android Instructions

Discoverability

Once your game is published, it is very important for people to be able to find it. The first days and weeks that your game is available to the public are critical. Sales of any given title can drop off quickly, and the speed of that drop partially depends on how many people know about the title. The people who want to download or buy your game need to know about it before they can do so. Prior to release, spend some time thinking about how to market your game. When uploading to the iTunes store, you’ll have an option to include a link to a YouTube video with your game listing. That’s free marketing—and you’d be silly not to take advantage of it by having a short, punchy video of your gameplay ready for linking by the time you get to this step.

The “walled garden” of Apple’s App Store can offer preferred placement to apps that show superior quality or innovative content.

Luis Levy on Creating a Successful Mobile Game Trailer:

Luis Levy is the co-founder of Novy Unlimited and the director of Novy PR—a public relations firm specializing in high-technology, mobile, and indie developers with clients such as Appy Entertainment, SRRN Games, PlayScreen, Swarm, Greenpeace, Shadegrown Games, and Woo Games. At Novy, Luis manages strategy, planning, media placement, speaking opportunities, and trade show bookings. Prior to Novy, Luis was an account executive at The Bohle Company—where he represented game and high-technology clients such as Spacetime Studios, The Voxel Agents, Muzzy Lane, and TimeGate Studios. Luis has also worked in advertising, sales, film and television editing, and as a game tester at Activision and Treyarch. He co-authored Play the Game: The Parent’s Guide to Video Games and Game QA & Testing with Jeannie Novak. Luis was born in São Paulo, Brazil and attended Fundação Armando Álvares Penteado (FAAP), where he received a B.A. in Film & Television.
Step 1: Make the Trailer
It’s hard to believe that some developers choose not to make a trailer. But it happens. While screenshots are easy and straightforward, trailers sound terribly complicated and time-intensive. Don’t let the extra workload deprive your game of a proper trailer. The consequences are often dire, starting with less coverage and ending with non-existing sales.

Step 2: Make Every Second Count
The only thing worse than not having a trailer is having a three-minute one. No one will watch a three-minute trailer for a mobile game. The ideal length is 45 seconds to a full minute.

Step 3: Make It Dynamic
We like trailers that show off different gameplay mechanics, characters, locations and atmosphere. You can’t have a minute of the same exact gameplay, over and over again. Think of the joy of driving on the Pacific Coast Highway versus the mind-numbing boredom of the Interstate.

Step 4: Make It Descriptive
Text-less trailers might look stunning but will often leave viewers with a big question mark. Is it a sequel or an update? Is it for Android or iOS? Is this a new feature? Is it free? This information must be in there. If the best you can do is list the most important information at the very end, with bullets, that’s still better than leaving everyone in the dark.

Step 5: Make It Real
Killzone 2 is famous for trying to pass CGI for actual gameplay. Don’t be tempted to make the same mistake. I know it’s easier to re-purpose the intro CGI or cut-scenes in the trailer, but you will lose your target audience in the process. Gamers watch trailers on YouTube and GameTrailers in order to evaluate a future purchase. They need to see how the game plays, not how great the cut-scenes are.

Step 6: Make Your Mark
Lush trailers can cost a lot of money. They usually have sweeping 3D art, post-processing, sharp HD visuals and booming soundtrack. This is awesome if you can pay for it, but it doesn’t mean a more “indie” trailer can’t compete. Be creative. Make the trailer an extension of your game. Give it an attitude. That’s the only way to fight the big boys, with their over-produced trailers and licensed music.

[This material originally appeared on the Novy PR web site at http://novypr.com/post/21939549035/trailers-win-the-youtube-wars-get-ink-make-it-a-hit]
Affinity Marketing

A good way of marketing your game is to find the people who will be most interested in purchasing it. If your game is about a train, for instance, there are dozens of train enthusiast web sites and online forums. Why not find out who’s in charge of those communities and let them know about your game? If you’re polite and succinct, you may get a mention on the web site’s home page; this puts your game in front of a large number of potential customers. Be considerate; don’t overdo it by being one of those overzealous developers that spam people weekly (or daily)!

Future Possibilities

Mobile games have changed drastically over the years and are sure to change even more in the future. What possibilities lie in store? One area that seems to be gaining popularity is augmented reality (AR), which attempts to meld what is happening in the real world into the game. At this point, it is difficult to find a phone that doesn’t have a camera in it, and mobile games are able to take advantage of the camera to bring the outside world into the game. For example, Star Wars Arcade: Falcon Gunner has an AR mode that lets you shoot Tie Fighters in your own living room. Another area that is currently expanding is voice recognition/personal assistant software such as Siri (iPhone). At this point, voice recognition is being adopted for a small number of titles such as Scribblenauts Remix on iPhone 4S.

Augmented Reality: Camera, Gyroscope & GPS

Augmented reality (AR) games use the camera to bring the real world into the game. The gyroscope is used to aid in navigation—and even the global positioning system (GPS), now built into most phones, is being used to map real-world places onto the game experience.
Going Mobile

Your first mobile game will be a blend of your own innovative ideas and old “hat tricks” that have produced quality gameplay through several generations of consoles. Just because a game type has been used before does not mean there isn’t a way to give it a fresh feel, a new twist. That blend of new and nifty and old and comfortable has carried a great many games to success. In the best “garage” spirit of the 1970s and 1980s, mobile games have re-opened the world of game development to solo agents and small teams. Tools such as GameSalad Creator are giving both budding and experienced game designers the chance to develop and sell games that might not find a home at a larger studio—or might be too daring to fit in with a publisher’s current lineup. The pre-coded elements and the drag-and-drop nature of Creator’s interface are designed to give you the power to release a game across multiple platforms—thus reaching the widest possible audience.
CHAPTER REVIEW EXERCISES

1. Create an idea for a mobile game. Consider theme, goals and target market. Don't worry about fleshing things out too much at this stage. Your goal is to create a concept in 1-3 paragraphs.

2. Odds are, your game fits cleanly into one of the more common game genres (platformer, puzzler, etc.). As you look through the different sample games that Creator has to offer, make a list of the genres. Are there any missing?

3. Will your game work better on a tablet or phone? Can you play it with one hand? Will the extra screen real estate on a tablet be a plus—or do you need to figure out a clever way to make use of the extra space?

4. Create some art assets for your game—a background, foreground elements, and a player character (which could be as simple as a steel ball or as elaborate as a fully armored knight). Be sure to size the artwork properly for use in your target platform.

5. Find a Creator template that is similar to the type of game you have been working on and open it. Look at the scenes, and go inside the actors—seeing what behaviors are attached. How do the pieces fit together to create the game?

6. Mobile games often take far fewer art assets than you might think. Between clever re-use, procedural elements such as particles, and coded-in elements such as palette stitching, you can make just a few images go a long way. Create a new project and import your images into it. Drag some actors onto the screen and start developing your game! When you have something you like, preview it. If it works, try installing one of the viewers onto a device (if one is available) and try playing your new game!

7. Think about ways you can promote your finished game. Come up with a few unique ways you might be able to promote your game based on its content. How could you make AR or Siri-like technology work in your game?